29 | 向前一步：万人规模企业的DevOps实战转型案例（上）

2019-12-26 石雪峰

DevOps实战笔记 进入课程 >

讲述：石雪峰
时长 17:49　大小 14.29M

你好，我是石雪峰。

“向前一步”这个名字，来源于 Facebook 的首席运营官谢丽尔·桑德伯格的一本书《向前一步：女性，工作及领导意志》。她在书中鼓励女性在职场中向前一步，勇于面对挑战，追求自己的人生目标。

我之所以选择用这个名字作为案例的标题，是因为在企业中，DevOps 转型并不是一件容易的事情，我们也需要有勇气向前迈出一小步，去承担这个使命。哪怕只是改变了一个小问题，也是转型过程中不可忽视的力量源泉。

在专栏最后的案例环节，我会用两讲给你介绍下微软这些年的 DevOps 转型故事，以及我在国内企业中的实践总结和经验。

今天，我们先从**管理实践**和**文化层面**入手，来看看这家传统的软件巨头是如何在经历了移动互联网时代的迷失之后，在容器云和 AI 时代再一次独占鳌头的。

微软的 DevOps 转型并不是一个突然的决定，**随着云服务的兴起，用户需求激增，对发布节奏的要求越来越高**。这些通过**需求的数量**就能体现出来。2016 年的用户需求数量比过去 4 年的总量还要多，到了 2017 年，需求数量达到了 2016 年的 2 倍，这就要求团队能够以更快的速度完成交付。

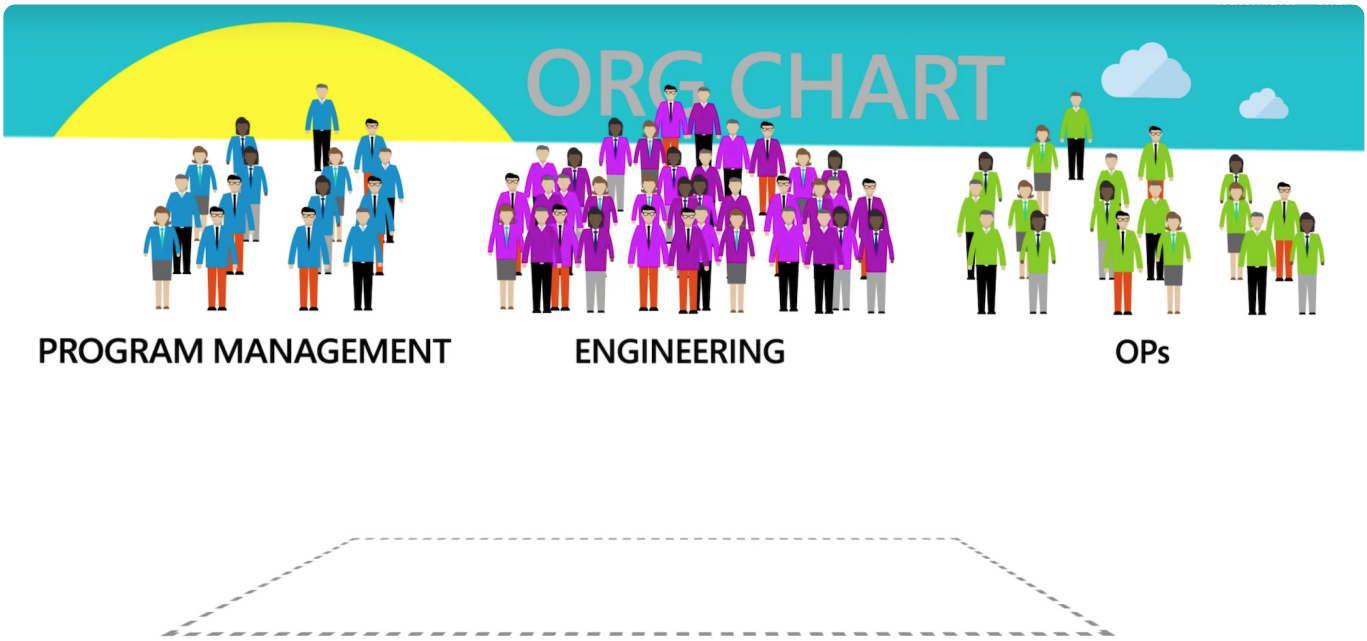
要知道，如果你期望的优化水平是提升 10% 的交付能力，那在原有的组织架构、流程规则下做局部优化就有可能实现。但是，如果要达到 200% 的优化效果，就需要做出巨大的改变了。

建立面向交付的特性团队

微软之前的组织架构跟很多公司一样，也是按照职能划分的，比如分为项目管理团队、开发团队、测试团队和运维团队。每个团队都比较封闭，部门墙的问题非常严重，给团队内部的协作效率造成了很大的影响。

为了应对需求交付的压力，微软首先进行了一次组织架构调整，**将开发团队和测试团队整合为工程团队**。于是，测试的职能在团队中消失了，转而变成了面向开发的开发工程师和面向

测试的开发工程师，他们和产品管理团队一起完成项目的敏捷推进。在敏捷的理念中，测试活动应该内嵌于开发环节之中，通过把两个部门整合起来，就完成了测试注入研发的工作。



虽然开发和测试团队融合到了一起，但是交付工作依然依赖于独立的运维团队来完成，这就造成了一个问题：**即便开发效率再高，如果运维能力跟不上，那也是没有意义的。**

于是，微软开启了第二次组织变革，这一次的核心是**构建特性交付团队，并赋予团队自治的能力。**

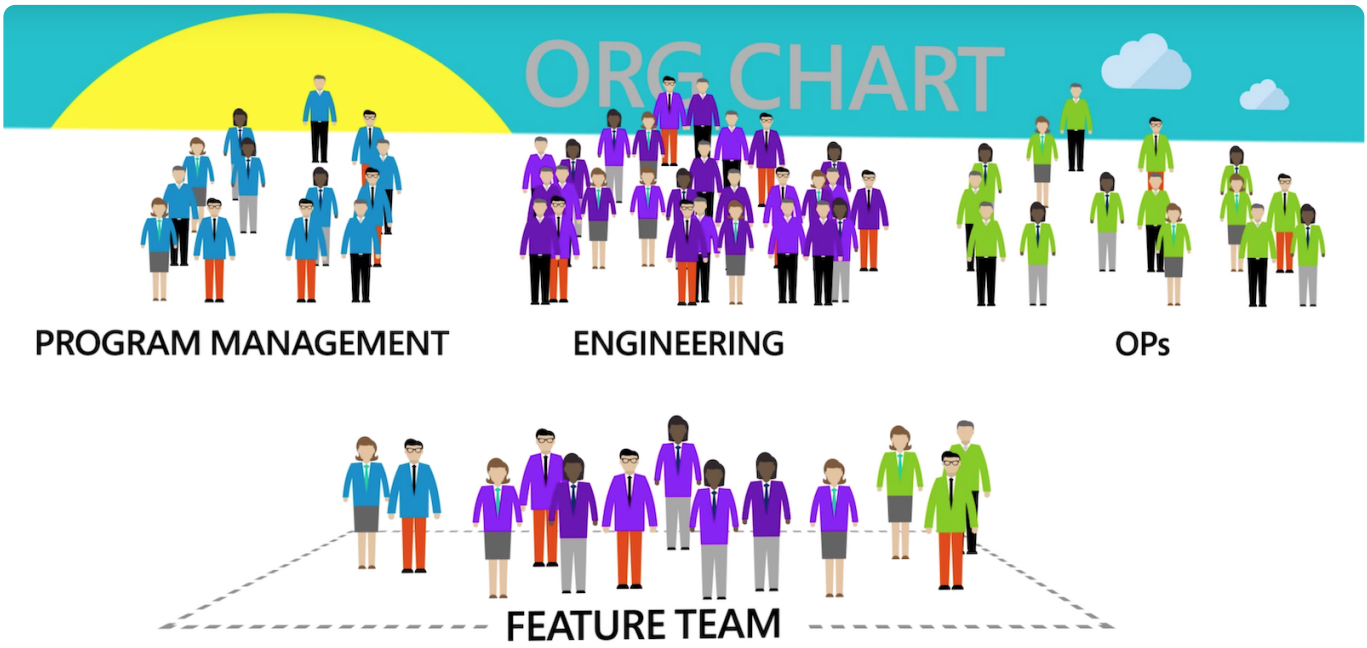
所谓的特性交付团队，就是我们常说的“全功能团队”，实际上，这就是把横向的按照职能划分的组织变成垂直跨职能的组织。这个团队中包含了要完成功能交付的所有角色（比如产品、开发、测试和运维），可以闭环地完成整个交付工作。

在这个过程中，微软引入了一种叫作**自组织团队**的形式。与传统的管理层自上而下安排组织的方式不同的是，员工可以自由地选择想要加入的特性团队。这种新的自由组队的方式为每个人都提供了学习新知识的机会。

你可能觉得，这么搞的话，组织不是乱掉了？高手都希望跟高手在一起，那剩下的同学怎么办呢？其实，我在国内的一家公司也见过类似的玩法，他们解决这个问题的核心方法就是“**传帮带**”模式。

比如，一个职能依赖某种特殊技能，但这种技能在团队内部非常稀缺，无论拥有这种特殊技能的这名成员去到哪个小队，剩下的组都会出问题。所以，这家公司就强制采用“老带新”的模式，也就是师傅对新人进行集中培训，给新人快速赋能。而且，这种“师徒关系”会长期存在，如果新人遇到什么问题，都可以请教师傅。当然，对于新人来说，也会有相应的考查机制。这种模式就有助于公司达成内部成员互相学习的目标。

根据数据统计，虽然只有不到 20% 的员工选择了岗位变化，但是这种方式却给 100% 的员工提供了选择的可能性。对于一家官僚政治出名的公司来说，这就可以大大地调动员工的积极性。



实际上，特性交付团队还有几个显著的特征：

1. 拥有团队独立的办公空间，大家都坐在一起，在沟通时基本可以靠“吼”；
2. 一般由 10～12 个团队成员组成；
3. 具有明确的工作目标和职责；
4. 为了保证稳定性，一旦组队成功，未来的 12～18 个月不再改变；
5. 自己管控特性向生产环境部署；
6. 团队自治。

无独有偶，国内某大型公司在推进 DevOps 转型的初期，也做了类似的事情。为了加速研发和运维的融合，它们将一个大的应用运维团队拆分到了各个业务线里面。

不仅如此，研发开始向全栈转型，承接运维工作，而运维自身的工作释放出去后，就要求团队进行能力升级。运维团队需要具备研发能力，来不断地开发和优化运维工具，以降低研发、运维的成本。

这个过程说起来轻松，但实际在做的时候，就需要非常强的组织执行力，甚至还需要高层背书，自上而下地贯彻这样的要求。转型的过程对于每个人来说都是很痛苦的，但也只有经过这样剧烈的变革，才让 DevOps 转型成为了现实，而不仅仅只是说说而已，或者只是在几个小部门之间搞来搞去。

我经常说一句话：“**想在不改变流程的前提下，实现企业的 DevOps 转型是不现实的。**”至于团队的组织架构是否要调整，还是由交付效率来决定的。

转型初期的引入工具和推广阶段对组织的冲击力没有那么大，但是，当转型到达了“深水区”之后，**组织的变革**就成了一个非常现实的问题。

根据“康威定律”，**一个团队交付的系统结构和他们的组织结构是相同的。**其实换个角度来说，软件交付的流程也是跟当前的组织结构保持一致的。只要有一个独立的测试团队，就总会有一个独立的测试阶段。而正是因为这样的一个个阶段，才带来了内部协作的部门墙和效率瓶颈，而这都是 DevOps 转型需要考虑的事情。

自组织敏捷团队

回到案例部分，为了促进特性交付团队的自治，微软在敏捷开发计划方面也进行了一定的调整。

首先，按照不同的维度，他们分为四种计划。

迭代维度：设定为 3 周一个迭代；

计划维度：包含了 3 个迭代；

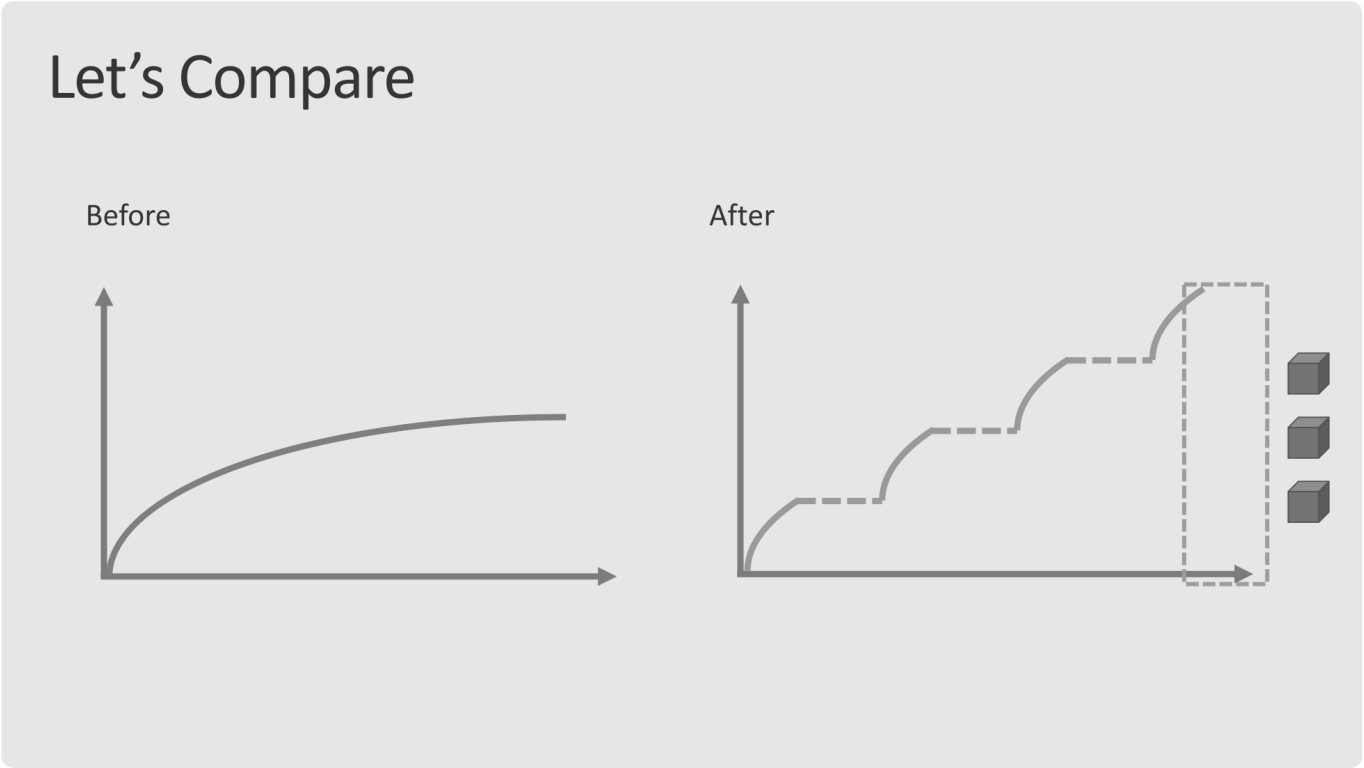
Season 维度：6 个月包含了两个计划周期；

Scenario 维度：长达 18 个月的远景图。

其中，管理层负责规划长期目标和全景图，也就是回答“我们要去哪里”的问题；而中短期目标，也就是迭代和计划，由自组织团队自行决定，这回答的就是“我们如何去到那里”的问题。

交付节奏按照迭代来进行，每 3 周的迭代会有一部分价值产出。随着迭代的不断推进，再逐步更新、优化计划目标，并反馈给长期规划，进行互动和调整。也就是说，6～18 个月的长期计划并不是一成不变的，团队会基于每个迭代和计划的交付增量以及用户的反馈进行调整，建立起一种“**计划 - 交付 - 学习**”的闭环路径，不断地校准产品目标和整体方向，保证长期规划的有效性，从而规避了原本瀑布模式下的在项目初期决定未来开发路径的潜在问题。

毕竟，在这个快速变化的时代，谁也无法保证你的计划是一成不变、永远有效的。

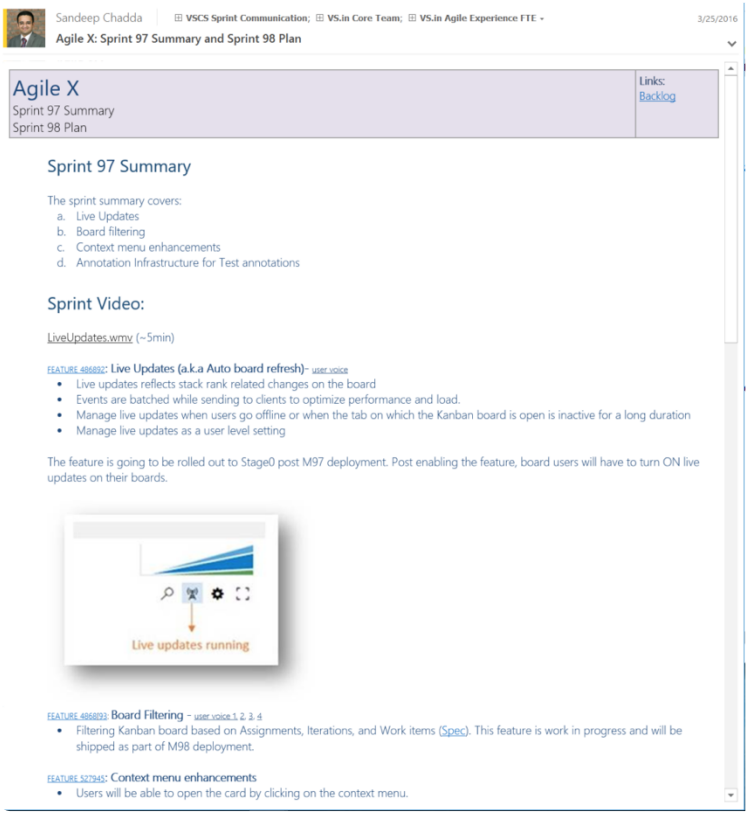


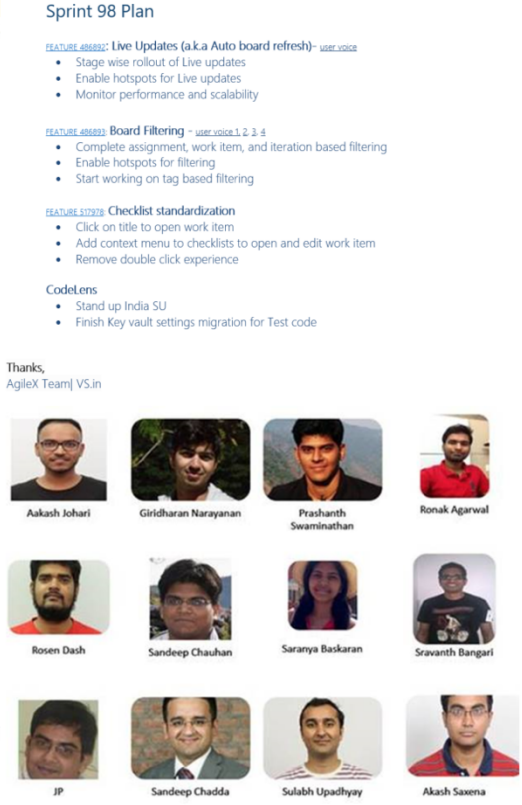
现在，特性交付团队的迭代和计划是由自己来决定了，但是你别忘了，每个成功的项目都需要成百上千人的协作。那么，如何保证团队目标的一致性和互相的配合度呢？

微软引入了三种实践方法，分别是**迭代邮件**、**团队交流**、**体验评审**。

1. 迭代邮件

在每个迭代的开始和结束时，团队都会发出迭代计划和状态邮件。在邮件中，除了明确本次迭代的特性完成情况以及下个迭代的交付计划之外，为了帮助其他团队成员更好地了解这个迭代的功能，他们还将这些功能录制成视频附在邮件里。不仅如此，**待办事项的列表**和**看板状态**也都以链接的形式被附在了邮件中。





2. 团队交流

在每次迭代完成的时候，团队成员都要问自己三个问题：

下一步的待办事项中包含哪些内容？

有哪些技术债务的累积和非功能特性？

有哪些遗留问题？

团队中的每个成员都要亲自完成这项任务，这不仅是为了减少信息传递的损失，更重要的是建立一种仪式感，帮助大家更加理性地安排迭代计划。毕竟，一旦把任务安排好了，就要按时完成。

3. 体验评审

在分析需求之初，采用用户故事的形式，站在用户的角度，以场景化的方式来描述用户所处的现状，以及这个特性想要解决的问题。那么，不同团队的成员就可以站在用户的视角来实现这个功能。

特别有意思的是，微软在管理特性的时候，**会尽量保持对原始用户需求的关联**。他们会在特性旁边附上原始的用户需求。

很多时候，开发要处理的任务都是**被产品人员翻译过的用户需求**，而并非原始的用户需求，以至于在开发的时候，**我们并不知道要解决的核心问题是什么**。通过关联原始用户需求，每个人都能在开发、测试、交付的过程中，站在用户的视角来重新审视一下，我们交付的功能到底是不是用户想要的。

这些环节的变化带来了一系列积极的影响。

首先，团队成员的积极性大大提高。因为他们觉得自己是用户体验的首要负责人，他们有责任自己修复并解决用户的实际问题。

其次，团队无需再等待领导的规划。在符合整体项目计划的前提下，他们可以自行制定计划。

最后，计划的更新是由持续学习来驱动的。比如，团队会给用户经常使用的功能添加埋点，观察用户使用的数据情况，定期关注和解决用户反馈信息。

持续的增量交付和不断的反馈建议，也是现在保证产品需求有效性的最佳手段。毕竟，业务敏捷是 DevOps 的源头，如果业务自己对需求都没有明确的衡量方法，那么即便拥有了最强的持续交付能力，也是跟“蒙眼狂奔”一样。所以，想要推进 DevOps，**敏捷开发实践和需求价值分析**都是必不可少的要素。

在微软，为了促进有效反馈，他们的**度量体系**也很好，非常值得一说。对于微软来说，**获取用户的真实行为数据至关重要**。他们在建设指标体系的时候，出发点大多落脚在**考量哪些指标对业务衡量有直接作用**上，而不是衡量团队的产出以及个人的产出。

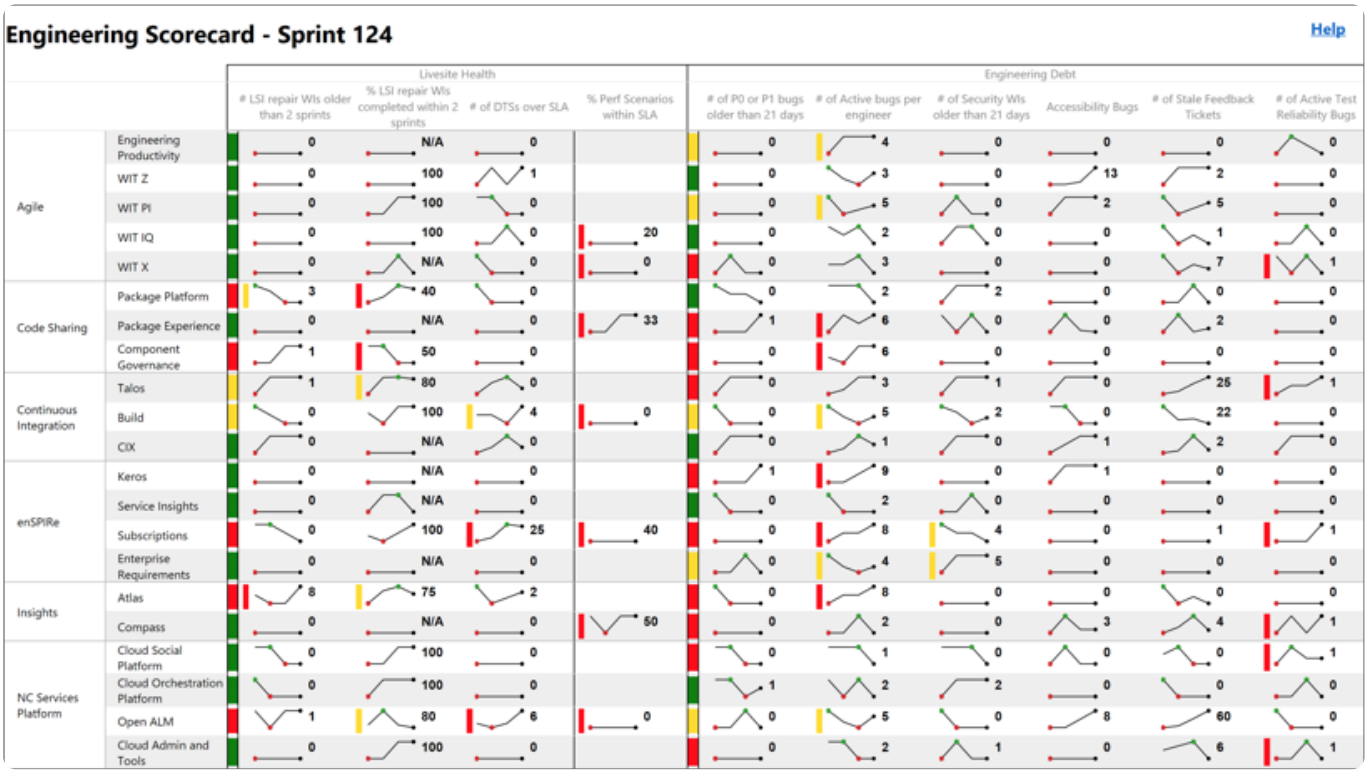
他们采用的指标包括以下几个方面：

使用维度：用户增长、用户满意度、特性交付情况等；

效率维度：构建时长、自测时长、部署时长等。

在线站点健康度：错误定位时长、用户影响时长、线上问题的遗留时长等。

但是，某些国内流行的指标却并没有被纳入绩效考核，比如**完成时长**、**代码行数**、**缺陷数量**等。



你可能会说，这也没什么特殊的啊，但是，你要知道，微软对于用户的关注不止如此。

我给你举个具体的例子。一般情况下，我们在度量系统可用性的时候，都是面向系统整体的，比如保证整体可用性达到 4 个 9，也就是在 99.99% 的情况下是可用的。但是，微软认为，系统可用性应该更进一步，**要以账号的维度来进行度量和统计**。

当我们站在系统整体的视角时，很多个人用户的行为就被整体数据掩盖了，也就是我们常说的“被平均”了。但是，如果站在账号的视角，也就是每一个用户的视角来看待这个问题的时候，我们就会发现，用户是真真切切地遇到了一些问题。

比如，某一个账号下服务不可用的情况出现频率比较高，那么，与其等着用户上网吐槽，倒不如提前跟用户取得联系，主动帮助他们解决问题。在联系用户的邮件中，不仅要清楚地描述团队观察到的客观情况，还要提供建议的解决方案。如果用户无法自主完成定位和修复，还可以通过邮件中的联系方式和团队取得联系，寻求进一步的帮助。

微软对用户的关注不仅体现在系统可用性的度量方面，在特性开关方面也是如此。

特性开关是一种比较常见的在运行时控制特性是否对外可见的技术手段。在微软的产品中，也大量地使用到了特性开关的技术，但他们的特性开关可以**细化到用户级别**，也就是可以将用户添加到或者移出列表中，从而控制每一个用户的可见特性。

这样一来，如果某些新特性影响了特定用户的使用，就可以通过这种方式处理，无需部署，直接将特性下线。这不仅有助于问题的快速解决，还提供了一种更加精细化的实验机制。与灰度发布相比，基于特性的发布也更加灵活。

团队转型从中型团队入手

在转型团队的选择方面，微软的经历带给我们的启示是，**尽量避免从大型团队开始入手**。

在 DevOps 转型的过程中，常见的思维方式是，先把企业内部最核心和最大的团队搞定。只要把最复杂的部分搞定了，其他中小团队的需求也就都可以满足了，他们会自然而然地跟上转型的节奏。

但是事实上，这些大团队往往都有一些独特的流程以及特殊的需求，对系统工具和流程的定制化程度较高，实现起来也最复杂，甚至对他们来说，转型工作的优先级并不是最高的，总会因为这样那样的需求导致转型工作一拖再拖。这对于转型工作来说，并不是一件好事。

所以，微软调整了他们的策略，采用了“middle-out”的方法，也就是**专注于中型团队**（40 到 100 人）。这些团队由于资源不像大团队那样充足，对外有充分的需求。而且，这种规模的团队可以快速地评估现状，收集团队的必要信息，而不是猜测他们到底需要什么。

通过持续细小的改进，帮助这样的团队做得更好，内部的传播让更多的团队主动联系他们并寻求帮助，从而建立了一个有效的持续改进循环。

总结

今天，我给你介绍了微软 DevOps 转型的上半部分内容。我们来简单总结一下。

为了满足快速交付的需求，他们打破组织的原有架构，建立了面向特性交付的跨职能组织；

通过团队自治，他们将计划分为短期目标和长期目标，短期目标（包括迭代和计划）都由特性团队来自主决定；

在度量方面，他们更加关注业务指标的表现。而且，无论是在系统可用性方面，还是在特性开关方面，他们都细化到了具体的用户级别，以保证每个用户的使用体验；

在选择转型团队方面，他们主动避开了最复杂的团队，而是从能够把握住的中型团队做起，积累成功经验，然后不断传播。

最后，我再提一点自己的想法。这两年来，在 DevOps 领域，特性的出镜率越来越高。因为特性是更加符合 DevOps 快速交付原则的需求颗粒度。所以，业界的各大公司在基于特性的需求管理、基于特性的分支策略、基于特性的发布和价值追踪策略等层面，都有很多实践和思考。比如，今年 CloudBees 公司发布的 SDM 产品就是基于特性维度的。

我相信，未来的 DevOps 也会朝着这个方向发展。打造一整套基于特性开发的研发模式，是一个值得我们花精力好好思考的点。

思考题

你认为，基于特性维度的开发和交付，有哪些流程、工具、规则是有效的呢？

欢迎在留言区写下你的思考和答案，我们一起讨论，共同学习进步。如果你觉得这篇文章对你有所帮助，也欢迎你把文章分享给你的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　28 | 迈向云端：云原生应用时代的平台思考

下一篇　30 | 向前一步：万人规模企业的DevOps实战转型案例（下）

精选留言 (3)

写留言

leslie

2019-12-26

最近一直在整理笔记，然后整理的过程中发现了自己的问题以及后续应当去做的努力。报了几门程序开发的课程-强化一下自己的代码功力。

　　老师现在一篇文章从看完到简单梳理差不多要1小时左右。今天的课程非微软的部分案例场景在上次大会听过-填鸭似的听了2天，真心觉得比上班还累啊。大企业的转变在某些方面都有些类似的，还是对于今天的课程做点补充吧。...

展开

作者回复: 我也补充一点，面向特性的交付团队，这个跟软件架构也有关联，很多公司不是不想按特性交付，而是软件架构决定的交付的节奏一定快不了，这也是为什么很多人在讲微服务的原因，当然拆服务不是个盲目的事情，我只想说康威定律真的是在很多地方都有印证，而逆康威定律也一样有效。

3

陈斯佳
2019-12-26

efficiency不等于effective。快，不一定对，方向很重要。有一个对方向不断矫正的反馈机制更重要。

作者回复: 有一本书叫做《Effective DevOps》，不是一般想象中的那种DevOps书籍，而更多的把视角落在的组织和人的层面上。在公司时间久了，就会发生一家公司能不能实现DevOps，跟工具的关系是最小的，人和流程，以及思维定式才是最大的问题。就好比两个老板要约一个会议，都是跟自己手下人说，你去找那边的老板约时间，那边的老板时间不妥当，也会让他的手下同步回来。其实想想，为啥两个老板不直接聊下呢，是因为没有工具吗，微信可方便的很呀，而这就是很多问题的根源所在吧。

1

maomaostyle
2019-12-30

老师可否解释下具体什么叫特性？这个概念如何定义？谢谢